# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-087271
(43)Date of publication of application : 02.04.1996

(51)Int.Cl. G10H 1/00
G10K 15/04
H03M 7/46

(21)Application number : 06-248875
(22)Date of filing : 17.09.1994

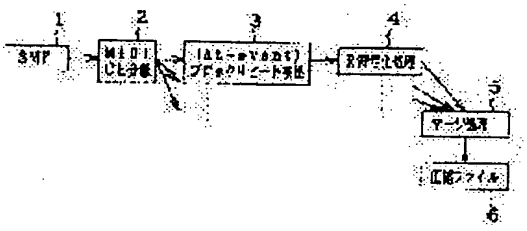
(71)Applicant : VICTOR CO OF JAPAN LTD
(72)Inventor : HIKAWA KAZUO
KUROIWA TOSHIO

## (54) COMPRESSION METHOD AND EXPANSION METHOD FOR PLAYING INFORMATION

(57)Abstract:

PURPOSE: To compress playing information in order to reduce the data amount of playing information which specifies the interval of sound constituting musical tone and the sounding time by deleting a repeated part so as not to overlap the repetition part when the existence of a repetition is detected.

CONSTITUTION: SMFs of many musics are stored in a memory. An SMF1 equivalent to one music specified by a control system is separated every channel by an MIDI CH separation block 2. A block 3 detects a (▵t+event) block repeat and when there exists a repetition in the playing information pattern, the block 3 detects the repetition. A block 4 performs an R(repeat) coding processing, deletes the repeated part detected by the block 3 so as not to overlap the repetition part and adds necessary signals. A merge processing block 5 merges the signals to which multi-channel R coding processings are performed into an original SMF format. Thus, ▵t is in integrated into one body and a compressed file 6 is produced.




LEGAL STATUS

[Date of request for examination] 26.03.1997

[Date of sending the examiner's decision of rejection] 27.06.2000

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁(JP)　(12)公開特許公報（A）　(11)特許出願公開番号

特開平８－８７２７１

(43)公開日　平成8年(1996)4月2日

| (51)Int. Cl.[6] | | 識別記号 | 庁内整理番号 | F I | 技術表示箇所 |
|---|---|---|---|---|---|
| G10H | 1/00 | 102 Z | | | |
| G10K | 15/04 | 302 D | | | |
| H03M | 7/46 | | 9382－5K | | |

審査請求　未請求　請求項の数７　FD　(全１６頁)

(21)出願番号　特願平6-248875

(22)出願日　平成6年(1994)9月17日

(71)出願人　000004329
日本ビクター株式会社
神奈川県横浜市神奈川区守屋町3丁目12番地

(72)発明者　飛河　和生
神奈川県横浜市神奈川区守屋町3丁目12番地　日本ビクター株式会社内

(72)発明者　黒岩　俊夫
神奈川県横浜市神奈川区守屋町3丁目12番地　日本ビクター株式会社内

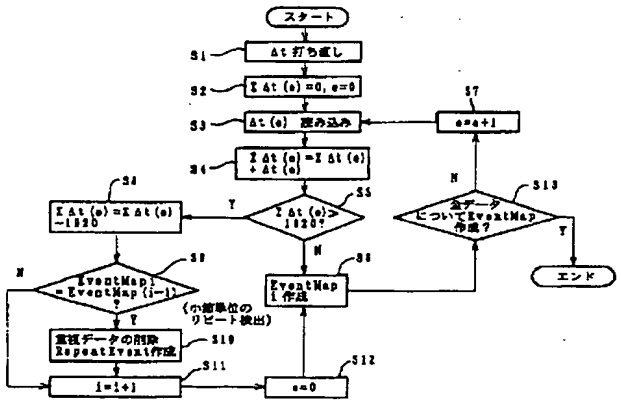
(74)代理人　弁理士　二瓶　正敬

(54)【発明の名称】演奏情報圧縮方法及び演奏情報伸張方法

(57)【要約】

【目的】　楽音を構成する音の音程と発音時間を指定する演奏情報のデータ量を大幅に削減すべく演奏情報を圧縮することのできる演奏情報圧縮方法と、かかる方法で圧縮された演奏情報を伸張して元の演奏情報を得ることのできる演奏情報伸張方法を提供する。

【構成】　演奏情報を所定時間単位のパート毎に分割し、パート毎の演奏情報を時系列順に比較し、時間軸方向の変化のパターンが２パート以上連続し、演奏情報に繰り返しがあるか否かを判断し、繰り返しがあると判断されたとき、繰り返しパート数を計算し、繰り返しのパートが重複しないよう削除し、削除される部分に、パターンの繰り返しが生じる旨を示す情報と、計数されたパターンの繰り返し回数を示す情報を付加する。再生時には、これらの情報を用いて圧縮された演奏情報を伸張する。他の実施例では複数パートにわたった演奏情報が繰り返される場合の圧縮と伸張が可能である。



1

【特許請求の範囲】

【請求項１】　楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、

前記分割するステップにて得られた前記パート毎の前記演奏情報を時系列順に比較し、前記演奏情報の時間軸方向の変化のパターンが２パート以上連続し、前記演奏情報に繰り返しがあるか否かを判断するステップと、

前記判断するステップにて繰り返しがあると判断されたとき、その繰り返しパート数を計数するステップと、

前記判断するステップにて繰り返しがあると判断されたとき、繰り返しのパートが重複しないよう削除するステップと、

前記判断するステップにて繰り返しがあると判断されたとき、前記削除する手段によって削除される部分に、パターンの繰り返しが生じる旨を示す情報と、前記計数するステップによって計数されたパターンの繰り返し回数を示す情報を付加するステップとを、

有する演奏情報圧縮方法。

【請求項２】　楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、

前記分割するステップにて得られた前記パート毎の前記演奏情報を時系列順に比較し、前記演奏情報の時間軸方向の変化のパターンが一部の演奏情報を除いて、２パート以上連続するとき、前記演奏情報に繰り返しがあると判断するステップと、

前記判断するステップにて繰り返しがあると判断されたとき、その繰り返しパート数を計数するステップと、

前記判断するステップにて繰り返しがあると判断されたとき、繰り返しのパート中、繰り返しではない異なる部分を残して、他の部分が重複しないよう削除するステップと、

前記判断するステップにて繰り返しがあると判断されたとき、前記削除するステップによって削除される部分に、パターンの繰り返しが生じる旨を示す情報と、前記計数するステップによって計数されたパターンの繰り返し回数を示す情報を付加するステップとを、

有する演奏情報圧縮方法。

【請求項３】　前記判断するステップが前記演奏情報の時間軸方向の変化のパターンを比較する際、音の音程と発音時間の照合において所定の誤差許容範囲を設け、比較精度を粗くするステップをさらに有する請求項１又は２記載の演奏情報圧縮方法。

【請求項４】　前記分割するステップにてパートに分割されて得られた前記各パート内において、所定の音程が連続し、繰り返されているか否かを判断する連続音程検出ステップと、

前記連続音程検出ステップにて繰り返しがあると判断されたとき、その繰り返し音程の情報出現回数を計数する同一音程計数ステップと、

前記連続音程検出ステップにて繰り返しがあると判断されたとき、繰り返しの音程の情報が重複しないよう削除する音程情報削除ステップと、

前記連続音程検出ステップにて繰り返しがあると判断されたとき、前記音程情報削除ステップによって削除される部分に、音程の繰り返しが生じる旨を示す情報と、前記同一音程計数ステップによって計数された音程の繰り返し回数を示す情報を付加するステップとを、

さらに有する請求項１乃至３のいずれか１つに記載の演奏情報圧縮方法。

【請求項５】　楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、

前記分割するステップにて得られたパート列中の先頭のパートからパート単位で順次データ比較を行い、実質的に同一のデータのパートに同一パート番号を付与しファイルＡを作成するステップと、

前記ファイルＡの先頭からパート番号を調べ、重複する番号が出現したときは、それを削除した新たなファイルＢを作成するステップと、

前記ファイルＡとファイルＢを先頭のパートからパート単位でパート番号の比較を行い、互いに相違するパート番号が出現したら、又は前記ファイルＡのパートと比較する前記ファイルＢのパートが存在しないときは、挿入すべきジャンプマークとタグの各々における識別のためのレベルをインクリメントし、前記ファイルＢ中の相違するパート番号の直前に前記レベルのインクリメントされたジャンプマークを挿入し、かつ前記ファイルＡ中の相違するパート番号と同一の前記ファイルＢ中のパート番号の直前に前記レベルのインクリメントされたタグを挿入するステップと、

前記ファイルＡ中の前記相違するパート番号以後と前記ファイルＢの前記タグの挿入された直後のパート以後とをパート単位でパート番号の比較を行い、互いに相違するパート番号が出現したら、前記挿入するステップと同様の挿入を行うステップとを、

有する演奏情報圧縮方法。

【請求項６】　楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、

当初内容のないファイルＣを作成するステップと、

前記ファイルＣ内の位置を示すファイルＣのポインタを前記ファイルＣの先頭に設定するステップと、

当初内容のないパートリストを作成するステップと、

変数ｉに前記ファイルＡの最初のパート番号を設定するステップと、

前記変数ｉが前記パートリストにあるか否かを判断するパート番号確認ステップと、

前記パート番号確認ステップで前記変数ｉで示されるパ

ート番号がないときは、前記ポインタが前記ファイルＣの末尾にあるか否かを判断するポインタ位置第１判断ステップと、

前記ポインタ位置が前記ファイルＣの末尾以外のときは、前記ポインタが現在ある位置にジャンプマークＪＵＭＰ（ｎ）を挿入し、次に前記ポインタを前記ファイルＣの末尾へ移動させ、その後前記ポインタの位置にタグＴＡＧ（ｎ）を挿入し、前記変数ｎを１つインクリメントするステップと、

前記ポインタ位置が前記ファイルＣの末尾であるときは、前記ファイルＣに前記ファイルＡの前記変数ｉで示されるパートの演奏情報を付加し、前記パートリストに前記変数ｉを付加し、前記ポインタを前記ファイルＣの末尾に移動させるステップと、

前記パート番号確認ステップで前記変数ｉで示されるパート番号があるときは、前記ポインタが前記ファイルＣの変数ｉで示されるパートの先頭にあるか否かを判断するポインタ位置第２判断ステップと、

前記ポインタ位置が前記ファイルＣの変数ｉで示されるパートの先頭以外のときは、前記ポインタが現在ある位置にジャンプマークＪＵＭＰ（ｎ）を挿入し、次に前記ポインタを前記ファイルＣの変数ｉで示されるパートの先頭位置へ移動させ、その後前記ポインタの位置にタグＴＡＧ（ｎ）を挿入し、前記変数ｎを１つインクリメントするステップと、

前記ポインタが前記ファイルＣの変数ｉで示されるパートの先頭であるときは、前記ポインタを次のパートの先頭位置へ移動させるステップと、

前記２つの移動させるステップ一方の後に、前記ファイルＡの中の次のパートを検索し、存在する場合は前記変数ｉに前記ファイルＡの次のパート番号を設定する次パート番号設定ステップとを、

有し、以後前記ファイルＡに次のパートがなくなるまで、前記パート番号確認ステップ以降、前記次パート番号設定ステップまでを繰り返し実行するようにした演奏情報圧縮方法。

【請求項７】　演奏情報の繰り返し部分を削除し、繰り返し部分を再生するために挿入されたジャンプマークとタグであってレベルの付されたものを含む圧縮演奏情報ファイルをメモリにロードするステップと、

前記メモリにロードされた前記圧縮演奏情報ファイルについて前記タグを検出するステップと、

前記メモリにロードされた前記圧縮演奏情報ファイルについて前記ジャンプマークをそのレベル順に検出するステップと、



前記ジャンプマークが検出されたとき、先に検出された前記タグのレベルと、その前記メモリのアドレスを指定して前記メモリからのデータの読み出し制御を行うステップとを有する演奏情報伸張方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は演奏情報の圧縮及び伸長に関し、特に楽音を構成する音の音程と発音時間を指定する演奏情報のデータ量を少なくするための圧縮と、一旦圧縮された演奏情報の伸長に関する。

【０００２】

【従来の技術】楽音を構成する音の音程と発音時間を指定する演奏情報として例えばＭＩＤＩ信号がある。ＭＩＤＩ信号はＳＭＦ（スタンダードＭＩＤＩファイル）という形式でメモリ等に蓄積されたり、伝送されたりする。かかるＭＩＤＩ信号は従来の光ディスク等の記憶媒体に記録されたデータを読み出す方式のカラオケ装置に代って通信カラオケといわれているシステムに用いられている。

【０００３】この通信カラオケはカラオケの音楽、歌詞、関連する画像（動画又は静止画）等を光ディスク等から出力された信号の情報の形態でそのまま送信すると、データ量が膨大であることから、楽音情報についてはＭＩＤＩ信号としてステーションから各ユーザーの使用場所に設けた端末へ送信し、歌詞については文字コードとして送信するもので、必要に応じて所望の曲のデータを配信するものである。かかる例として株式会社エクシング製の「ＪＯＹＳＯＵＮＤ」システムが知られている。なお、音符、信号よりなるメロディの楽譜データについて繰り返し部分を効率よく処理する技術が特公昭６１－９１６９７号公報に示されているが、これはＭＩＤＩ信号を前提とするものではなく、繰り返し等の記号を用いた楽譜が存在することを前提としたものであり、ＳＭＦで表現されるような既にある演奏情報を圧縮する技術ではない。さらに特開平４－１４７１９２号公報には楽譜の繰り返し部分をパターン化する技術が示されているが、十分なデータの圧縮は期待できない。

【０００４】上記従来の通信カラオケシステムをはじめ、これまでに提案されたシステムは次の表１のように分類される。

【０００５】

【表１】

5　　6

| ソース | | データ量 | 音の品質 |
|---|---|---|---|
| ＣＤ、ＬＤを再生するもの（実際の演奏・コーラスが記録されている） | | 膨大 | 最高 |
| MIDIによるもの | 演奏して入力したものにコーラスを付加 | 大 | 高 |
| | 演奏して入力(コーラス付加せず) | 中 | 中 |
| | 音符から機械的に入力(コーラス付加せず) | 小 | 低 |

【０００６】
【発明が解決しようとする課題】表１からわかるように、ＭＩＤＩ信号を用いた場合、ＣＤ、ＬＤ等から再生された音の品質とは同一ではないが、品質を向上させようとすればデータ量が増大することがわかる。データ量が多いと、ＩＳＤＮのような高速デジタル回線を用いなければリアルタイム通信が不可能となり、またデータの蓄積、保存の上でも必要メモリー容量が大きくなりコストの上昇につながってしまう。かかる問題は、通信カラオケシステムが一般家庭へ普及していくための障害となっている。また、通信上の問題とは別に、ハードディスク等の大容量メモリに多数の曲の演奏情報を蓄積しておく、所謂パッケージカラオケシステムにおいても、データ量の圧縮が求められている。
【０００７】音楽は繰り返しの芸術であると言われるように、演奏が一定のパターンで繰り返されることが多い。ところがＭＩＤＩ信号のような演奏情報では時間軸上に発生する次々のイベントの順列をそのままの形で扱っているので、一般的なデジタル信号の圧縮技術によるデータ量の圧縮を行うにとどまっていた。このため演奏情報を配信する通信カラオケやメモリに多数曲のデータを蓄積したパッケージ形式のカラオケシステム等では、データ量の圧縮・削減によるコスト低減、通信時間の短縮等が求められている。
【０００８】したがって、本発明は楽音を構成する音の音程と発音時間を指定する演奏情報のデータ量を大幅に削減すべく演奏情報を圧縮することのできる演奏情報圧縮方法と、かかる方法で圧縮された演奏情報を伸張して基の演奏情報を得ることのできる演奏情報伸張方法を提供することを目的とする。
【０００９】
【課題を解決するための手段】上記目的を達成するために本発明では、演奏情報が一定の時間単位で繰り返され、重複部分があることに着目し、かかる繰り返し部分を検出し、これを削除するとともに、再生時に圧縮情報を正しく再現できるよう、削除された繰り返し部分の位置等を示す信号を挿入するようにし、再生時には、この信号を用いて圧縮された演奏情報を伸張して、元の演奏情報を得るようにしたものである。
【００１０】すなわち本発明によれば、楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、前記分割するステップにて得られた前記パート毎の前記演奏情報を時系列順に比較し、前記演奏情報の時間軸方向の変化のパターンが２パート以上連続し、前記演奏情報に繰り返しがあるか否かを判断するステップと、前記判断するステップにて繰り返しがあると判断されたとき、その繰り返しパート数を計数するステップと、前記判断するステップにて繰り返しがあると判断されたとき、繰り返しのパートが重複しないよう削除するステップと、前記判断するステップにて繰り返しがあると判断されたとき、前記削除する手段によって削除される部分に、パターンの繰り返しが生じる旨を示す情報と、前記計数するステップによって計数されたパターンの繰り返し回数を示す情報を付加するステップとを、有する演奏情報圧縮方法が提供される。
【００１１】さらに本発明によれば、楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、前記分割するステップにて得られた前記パート毎の前記演奏情報を時系列順に比較し、前記演奏情報の時間軸方向の変化のパターンが一部の演奏情報を除いて、２パート以上連続するとき、前記演奏情報に繰り返しがあると判断するステップと、前記判断するステップにて繰り返しがあると判断されたとき、その繰り返しパート数を計数するステップと、前記判断するステップにて繰り返しがあると判断されたとき、繰り返しのパート中、繰り返しではない異なる部分を残して、他の部分が重複しないよう削除するステップと、前記判断するステップにて繰り返しがあると判断されたとき、前記削除するステップによって削除される部分に、パターンの繰り返しが生じる旨を示す情報と、前記計数するステップによって計数されたパターンの繰り返し回数を示す情報を付加するステップとを、有する演奏情報圧縮方法が提供される。
【００１２】さらに本発明によれば、楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップ

と、前記分割するステップにて得られたパート列中の先頭のパートからパート単位で順次データ比較を行い、実質的に同一のデータのパートに同一パート番号を付与しファイルＡを作成するステップと、前記ファイルＡの先頭からパート番号を調べ、重複する番号が出現したときは、それを削除した新たなファイルＢを作成するステップと、前記ファイルＡとファイルＢを先頭のパートからパート単位でパート番号の比較を行い、互いに相違するパート番号が出現したら、又は前記ファイルＡのパートと比較する前記ファイルＢのパートが存在しないときは、挿入すべきジャンプマークとタグの各々における識別のためのレベルをインクリメントし、前記ファイルＢ中の相違するパート番号の直前に前記レベルのインクリメントされたジャンプマークを挿入し、かつ前記ファイルＡ中の相違するパート番号と同一の前記ファイルＢ中のパート番号の直前に前記レベルのインクリメントされたタグを挿入するステップと、前記ファイルＡ中の前記相違するパート番号以後と前記ファイルＢの前記タグの挿入された直後のパート以後とをパート単位でパート番号の比較を行い、互いに相違するパート番号が出現したら、前記挿入するステップと同様の挿入を行うステップとを、有する演奏情報圧縮方法が提供される。

【００１３】さらに本発明によれば、楽音を構成する音の音程と発音時間を指定する演奏情報を受けて、前記演奏情報を所定時間単位のパート毎に分割するステップと、当初内容のないファイルＣを作成するステップと、前記ファイルＣ内の位置を示すファイルＣのポインタを前記ファイルＣの先頭に設定するステップと、当初内容のないパートリストを作成するステップと、変数ｉに前記ファイルＡの最初のパート番号を設定するステップと、前記変数ｉが前記パートリストにあるか否かを判断するパート番号確認ステップと、前記パート番号確認ステップで前記変数ｉで示されるパート番号がないときは、前記ポインタが前記ファイルＣの末尾にあるか否かを判断するポインタ位置第１判断ステップと、前記ポインタ位置が前記ファイルＣの末尾以外のときは、前記ポインタが現在ある位置にジャンプマークＪＵＭＰ（ｎ）を挿入し、次に前記ポインタを前記ファイルＣの末尾へ移動させ、その後前記ポインタの位置にタグＴＡＧ（ｎ）を挿入し、前記変数ｎを１つインクリメントするステップと、前記ポインタ位置が前記ファイルＣの末尾であるときは、前記ファイルＣに前記ファイルＡの前記変数ｉで示されるパートの演奏情報を付加し、前記パートリストに前記変数ｉを付加し、前記ポインタを前記ファイルＣの末尾に移動させるステップと、前記パート番号確認ステップで前記変数ｉで示されるパート番号があるときは、前記ポインタが前記ファイルＣの変数ｉで示されるパートの先頭にあるか否かを判断するポインタ位置第２判断ステップと、前記ポインタ位置が前記ファイルＣの変数ｉで示されるパートの先頭以外のときは、前記ポインタが現在ある位置にジャンプマークＪＵＭＰ（ｎ）を挿入し、次に前記ポインタを前記ファイルＣの変数ｉで示されるパートの先頭位置へ移動させ、その後前記ポインタの位置にタグＴＡＧ（ｎ）を挿入し、前記変数ｎを１つインクリメントするステップと、前記ポインタが前記ファイルＣの変数ｉで示されるパートの先頭であるときは、前記ポインタを次のパートの先頭位置へ移動させるステップと、前記２つの移動させるステップ一方の後に、前記ファイルＡの中の次のパートを検索し、存在する場合は前記変数ｉに前記ファイルＡの次のパート番号を設定する次パート番号設定ステップとを、有し、以後前記ファイルＡに次のパートがなくなるまで、前記パート番号確認ステップ以降、前記次パート番号設定ステップまでを繰り返し実行するようにした演奏情報圧縮方法が提供される。

【００１４】さらに本発明によれば、演奏情報の繰り返し部分を削除し、繰り返し部分を再生するために挿入されたジャンプマークとタグであってレベルの付されたものを含む圧縮演奏情報ファイルをメモリにロードするステップと、前記メモリにロードされた前記圧縮演奏情報ファイルについて前記タグを検出するステップと、前記メモリにロードされた前記圧縮演奏情報ファイルについて前記ジャンプマークをそのレベル順に検出するステップと、前記ジャンプマークが検出されたとき、先に検出された前記タグのレベルと、その前記メモリのアドレスを指定して前記メモリからのデータの読み出し制御を行うステップとを有する演奏情報伸張方法が提供される。

【００１５】

【実施例】以下図面と共に本発明の好ましい実施例について説明する。図１は本発明の演奏情報圧縮方法の第１実施例を示すフローチャートである。図１のフローチャートについて説明する前に、本発明の適用される例として通信カラオケシステムについて図２、図３を用いて説明する。図２は本発明の演奏情報圧縮方法により、カラオケ情報としてのＭＩＤＩ情報を圧縮してステーション（ホスト）から各端末へ送信する通信カラオケシステムの一例を示す模式ブロック図である。図３はステーションが圧縮しないデータと、本発明の演奏情報圧縮方法により事前に圧縮してあるデータの双方を送信／蓄積ファイルＡ、Ｂの２種類用意しておき、端末や回線の能力に応じて２つのファイルを使い分ける構成例を示す模式ブロック図である。どちらのファイルから読み出して送信するかの判断は、通信回線（ネットワーク）がＩＳＤＮ等の高速回線か一般公衆回線かを判断したり、端末側の伸張能力を検出することにより行う。なお、図２、図３いずれのシステムにおいても、端末側のハードディスク等のメモリにパッケージとして予め数千曲分のカラオケ情報を本発明の演奏情報圧縮方法により圧縮したＭＩＤＩ情報として蓄積しておくことができる。この場合、新譜やリクエスト頻度の低い曲で端末のメモリに蓄積され

ていない曲の情報を通信回線を用いてステーションから端末へ送信することができる。

【００１６】図４は１つの曲の演奏情報を記述したＳＭＦを圧縮して圧縮ファイルを作成する手順の全体概念を示すブロック図である。多数曲分のＳＭＦは例えばハードディスク等のメモリに記憶されており、この内、図示しない制御系により指定された１曲分のＳＭＦ１がＭＩＤＩ　ＣＨ（チャンネル）分離ブロック２にてチャンネル毎に分離される。すなわち演奏すべき楽器（音源）に対応するチャンネル毎に以下の処理は行われる。ブロック３は（△ｔ＋ｅｖｅｎｔ）ブロックリピートを検出するものである。すなわち、後述する演奏情報のパターンに繰り返し（リピート）があるとき、これを検出するものである。次のブロック４はＲ（リピート）符号化処理を行うもので、ブロック３で検出された重複部分について重複しないよう削除して、かつ必要な信号を付加する処理が行われる。次のマージ処理ブロック５は多チャンネル分のＲ符号化処理の行われた信号を元のＳＭＦ形式のように合体するものであり、その結果△ｔ（デルタタイム）が一本化されて、圧縮ファイル６が作られる。

【００１７】図５はＳＭＦによる記述から繰り返し部分を検出し、重複部分を削除して、圧縮した結果得られる圧縮ファイルを作成する場合のデータ列の変化態様を示す図である。図５中“ＥＶＥＮＴ”はＭＩＤＩ信号の“イベント”を示し、“Ｔ”は“△ｔ”と同様の時間情報を、“Ｒ”は“リピート（繰り返し）”を示している。図６は図５に示した情報“Ｔ”又は“Ｒ”のフォーマットを示している。このデータは図示のように８ビット構成であり、各ビットは図示の内容を示す。図７は１つの例として、繰り返し部分のある楽譜を示している。この例では第２小節と第３小節が同一のパターンとなっている。

【００１８】ここで図１のフローチャートに戻って本発明の演奏情報圧縮方法の第１実施例について説明する。このフローは図４で示したブロック３とブロック４に相当する。ステップＳ１にてチャンネル単位で先頭から△ｔで表記し、ステップＳ２で△ｔ（ｅ）の累計であるΣ△ｔ（ｅ）と個々のイベントを示す変数ｅをそれぞれ０にイニシャライズする。ステップＳ３では△ｔ（ｅ）を読み込み、ステップＳ４では△ｔ（ｅ）だけインクリメントし、ステップＳ５ではΣ△ｔ（ｅ）が１９２０を超えたか否かを判断する。この数値１９２０は４分音符１個あたりのティクス（Ｔｉｃｋｓ）数を４８０とし、１小節が４分音符４個分の長さ（すなわち拍子記号が４／４）とした場合の１小節の長さに相当する。なおＴｉｃｋはＳＭＦにおける最小時間単位である。このステップＳ５は演奏情報であるＳＭＦを所定時間単位のパート（この例では１小節）毎に分割するものである。ステップＳ５で１小節が経過するまでの間にステップＳ６を繰り返し実行し、イベントマップ（ＥＶＥＮＴ　ＭＡＰ）ｉを作成する。イベントマップｉはパート毎のイベントの集合である。ステップＳ７ではｅを１つインクリメントする。１小節が終了するとステップＳ８で△ｔの累計から１９２０を減じ、次のステップＳ９で前の小節（ｉ－１）のイベントマップ（ｉ－１）と、現在の小節ｉのイベントマップｉの内容が一致するか否かを判断する。一致しないときはステップＳ１１でｉを１つインクリメントする、一致するときはステップＳ１０で重複データの後ろの方を削除し、リピートイベント（Ｒｅｐｅａｔ　Ｅｖｅｎｔ）を作成する。リピートイベントは図５に示す“Ｒ”に相当し、図６のデータフォーマットを有する。図６で（ＩＤ１，ＩＤ０）＝（０，０）及び（ＩＤ１，ＩＤ０）＝（０，１）の場合がＴ、（ＩＤ１，ＩＤ０）＝（１，０）及び（ＩＤ１，ＩＤ０）＝（１，１）の場合がＲである。単純リピート及び区間指定リピートはチャンネル単位で行い、チャンネルはリピートバイトの直前のイベントに基づいて決定される。図６の－△ｔｓはマイナス△ｔ－ｓｔａｒｔ、－△ｔｅはマイナス△ｔ－ｅｎｄの意味であり、－△ｔｓ及び－△ｔｅはマイナスを取って絶対値で表現している。なお、ステップＳ１０では、リピート検出回数を計数し、図６中のリピートイベント中のリピート回数として記述する。ステップＳ１０の後はステップＳ１１が実行される。ステップＳ１１の後にはステップＳ１２にてｅを０としてステップＳ６へ行く。なお、ステップＳ１３では全データについてイベントマップが作成されたか否かを判断し、ＹＥＳならこのフローを終了する。

【００１９】図１のフローによって図７の譜例に示される曲の演奏情報を処理すると、イベントマップ同志の比較から、同一の音程、同一の発音時間を示すパートの繰り返しとして、第２小節と第３小節が検出される。図１の実施例では演奏情報の時間軸方向の変化のパターンが２パート以上連続したとき、すなわちイベントマップの内容が完全に一致したとき、繰り返しがあると判断しているが、他の例として、部分的な一致を検出したときも、繰り返しとみなし、異なる部分はそのまま記述するようにすることもできる。

【００２０】さらに、パート同志の一致の有無の検出精度を粗くし、すなわち音程と発音時間の照合において所定の誤差許容範囲を設けるようにしてもよい。このようにすることにより、わずかなノイズや演奏ミスにより、本来繰り返し部分であるのに、非同一パターンと認識されてしまうことを防止でき、結果として繰り返しと認識されるパート数を増加させることができ、圧縮効率を高めることができる。さらに、演奏上の時間軸のゆらぎ、あるいは演奏の強さのゆらぎなどの許容範囲を演奏再現時（再生時）にあまり影響を与えない程度に広げることにより、圧縮効率を高めることもできる。また、パート同志の比較の他に、同一パート内で所定の音程が繰り返し出現するような場合、例えばドラムなどの打楽器の演

奏情報の場合、これを検出して重複しないよう削除し、図１で示した処理と同様の処理にて音程単位で記述することもできる。本発明では所定時間単位のパート毎の検出が行われるが、この１パートは必ずしも１小節と同一である必要はなく、その数分の１としたり、数倍とすることもできる。さらに、パートの時間を２種類以上設け、例えば１小節単位の検出を行いつつ、その２分の１の時間単位でも繰り返しの検出を行うようにすることもできる。

【００２１】図１に示した第１実施例は、連続するパートの内容が一致している場合に繰り返しであると判断しているが、いくつかの小節にわたったメロディーが繰り返される場合も多い。このような場合にも圧縮可能な手法について第２実施例と第３実施例で説明する。　図２３は圧縮前のＳＭＦで８つのパートからなるものを模式的に示した図である。図２３において、第２番目乃至第４番目のパートと第５番目乃至第７番目のパートが重複している。したがって冗長性を除くために、考えられる１つの手法として、重複する３つのパートを削除すると図２４に示すように５つのパートのみとなる。この場合、「先頭から（ｔ１＋ｔ２＋ｔ３＋ｔ４）の時間だけ再生したら、先頭からｔ１経過時点までジャンプする」という内容を記述しておくことになる。すなわち各パートの△ｔの総和をそれぞれ計算しておき、再生開始時点からの経過時間によって再生位置を再設定するような記述を設けている。しかし、この方法によると、パートの再編集の必要性が生じた場合、記述部を同様に編集しなくてはならない。また記述が複雑であるほど時間の再計算も複雑化してしまう。

【００２２】前述の特開昭６１－９１６９７号公報では、楽譜の省略記号をコード化し、音符情報と共に記憶し、再生時に省略記号に従って制御を行うようにしている。これによれば冗長性の問題と再編集の問題は解決するものの、現代の楽曲は構成が複雑化しており、省略記号を単にコード化しただけでは楽曲構成を正しく表記できない場合が生じてくる。図２５に示した例のように、Ｄ．Ｓ．（ダルセーニョ）によって戻って再度演奏する場合に繰り返しを省くことがよく行われるが、このように注釈によって指示される選択的な演奏情報を表記することは上記公報の技術では不可能である。また、上記公報の技術は前述のように繰り返し等の記号を用いた楽譜を前提としいる。

【００２３】したがって本発明の演奏情報圧縮方法の第２実施例及び第３実施例では、複数回重複して記述される同一パートを含み直列状に記述されている演奏情報に対して、演奏情報と識別可能なジャンプマークとタグを設け、楽曲の繰り返しを示すことによって、少ない情報量で記述するようにしている。また、ジャンプマークに条件値（レベル）を含めることによって複雑な演奏情報を記述することができる。

【００２４】図１３はジャンプマークとタグの構成を示す図である。いずれも、非演奏情報ＩＤ、チャンネルＩＤ、レベルＩＤを有している。レベルＩＤは同一マークの相互間の識別を行うか、又は順序を示すものである。なお第２実施例と第３実施例は、共に複数のパート（各実施例では小節）にわたる演奏情報が繰り返されるときに、繰り返しを削除してジャンプマークとタグを挿入し、結果として同じ圧縮ファイルを作るものである。第２実施例は元のファイルＡの他に重複のないファイルＢを一旦作成する必要がある点で後述する第３実施例より手順が多い。しかし、圧縮の手順を理解するためには適当であるので、まず第２実施例について説明し、その後、より利用価値の高い第３実施例について説明する。図１４と図１５は同一パートの検出と削除の概念を示す図である。図１４の１部の楽譜に示すように７小節と省略記号で記述された曲を展開すると図１４の下部に示すようなものとなる。なお丸印の中の数字は便宜的に記入したものであって、どの小節とどの小節の演奏情報が同一であるのかは、この時点ではわかっていない。

【００２５】図１５は図１４の下部に示されたような演奏情報があるとき、このデータ列から重複部分を検出する手法を示す図である。演奏情報は小節単位で時間軸上に図１５の上段に示すように並んでいる。ここで先頭の小節ｘから以降の小節に対してデータの比較を行う。一致したなら若い順にパート番号（この場合１）をふる。演奏情報の作成がパートのコピーによって行われない場合は時間軸方向の揺らぎ等が含まれるため、データのファイリングやクォンタイズ、認識的手法が用いられる。このようにしてパート番号のふられた情報を納めたファイルをＡ、重複したパートを削除した情報を納めたファイルをＢとする。図１６はこの２つのファイルＡ、Ｂと本発明の演奏情報圧縮方法の第２実施例により最終的に作成される圧縮ファイルを示している。両方のファイルＡ、Ｂを読み進め、異なるパートが出現したら、レベルｎをインクリメントしファイルＢにおいて、ジャンプマークＪ（ｎ）を挿入しファイルＡの次に出現するパートに戻ってタグＴ（ｎ）を挿入する。

【００２６】レベルｎはジャンプマークＪ及びタグＴのそれぞれにおいて相互識別をするためのものであり、自然数で１、２、３…とインクリメントする。図１６の下方には、ジャンプマークＪとタグＴを挿入する手法が示されている。すなわち、ファイルＡとファイルＢを先頭のパートからパート単位でパート番号の比較を行い、互いに相違するパート番号が出現したら、又はファイルＡのパートと比較するファイルＢのパートが存在しないときは、挿入すべきジャンプマークとタグの各々における識別のためのレベルｎをインクリメントし、ファイルＢ中の相違するパート番号の直前にレベルのインクリメントされたジャンプマークを挿入し、かつ前記ファイルＡ中の相違するパート番号と同一のファイルＢ中のパート

番号の直前にレベルのインクリメントされたタグを挿入する。ファイルＡ中の相違するパート番号以後とファイルＢのタグの挿入された直後のパート番号とをパート単位でパート番号の比較を行い、互いに相違するパート番号が出現したら、上記の挿入動作と同様の挿入が行われる。

【００２７】図１７及び図１８は上記第２実施例を用いて複雑な繰り返しのある曲の演奏情報を圧縮することができる例を示す図であり、図１７はかかる複雑構成の曲の例を示し、図１８は圧縮前のファイルＡ、Ｂと圧縮ファイルを示す図である。

【００２８】ここでＭＩＤＩ信号情報ファイルとしてのＳＭＦとＳＭＦを変形して上記ジャンプマーク、タグを挿入してゆく手法について説明する。ＳＭＦは以下のものを含む。

・ＭＩＤＩイベント　：　演奏情報
・ＭＥＴＡイベント　：　非演奏情報

どちらのイベントも時間情報△ｔを持っているので、ＭＥＴＡ（メタ）イベントは、ＭＩＤＩイベントとの相対的な時間的位置を指定して、挿入することができる。また、ＭＥＴＡイベントは、これが持つデータフォーマットを規定することができる。通常のＳＭＦでは、パートの概念はないが、パートの先頭であるという識別子（ＰＡＲＴ　ＩＤ）を持ったメタイベントを定義することによって、ＳＭＦ中にパート番号を記述することができる。

【００２９】次にパートの定義について説明する。例えば１小節に相当する各々のパートは図１５に示されるようにパート番号により指定される。したがって各パートはＭＥＴＡイベントから始る、ＳＭＦ中の一連のＭＩＤＩイベント群である。このＭＥＴＡイベントを、ＭＥＴＡ（ＰＡＲＴ，ｘ）と略記する。

ＭＥＴＡ（ＰＡＲＴ，ｘ）→△ｔ：ＭＥＴＡ　ＩＤ：ＰＡＲＴ　ＩＤ：ＰＡＲＴＮＯ　ｘ　とする。比較のために、ＭＩＤＩイベントのＮｏｔｅ　ｏｎ（打鍵）は、Ｎｏｔｅ　ｏｎ→△ｔ：ＭＩＤＩ　Ｎｏｔｅ　ｏｎ　ＩＤ：ＭＩＤＩチャンネル：音程：音量

ＳＭＦの構成は図１９に示すようになっている。

【００３０】ＭＥＴＡ（ＰＡＲＴ，ｘ）の挿入位置は、自由であるが、繰り返しの検出に用いるので、繰り返しが予想される単位以下で挿入するのが良い。例えば、図２０に示すような楽譜をもとにＳＭＦを作成する場合、作成者は、“Ａ，Ｂ，Ｃ，Ｄ，Ｅ”それぞれのＭＩＤＩ演奏情報を作成し、実際の演奏手順に従って、データのコピーペースト等の手法で“Ａ　Ｂ　Ｃ　Ｄ　Ａ　Ｂ　Ｃ　Ｅ”となったＭＩＤＩ演奏情報列をＳＭＦに収めるが、楽譜上の繰り返し記号と、ＭＥＴＡ（ＰＡＲＴ，ｘ）の入れ方の規約を作ることによって、図２１に示すようなＳＭＦを作成することが可能である。

【００３１】あるいは作成者が以上のような、作成手順に全く関知しない通常のＳＭＦを作成する場合、繰り返し部分を自動的に検出するためには、ＳＭＦを受け取った側で１小節＝１パートとなるように、ＭＥＴＡ（ＰＡＲＴ，ｘ）を挿入するのが良い。ＳＭＦ中の小節の切れ目は既にＳＭＦフォーマット中に定められている。ＭＥＴＡ（ｔｉｍｅ　ｓｉｇｎａｔｕｒｅ）等の情報を、ＳＭＦから読み出すことによって、簡単に知ることができる。前出の例によれば、ＭＥＴＡ（ＰＡＲＴ，ｘ）が挿入されたらＳＭＦは図２２に示すようになる。どちらの場合でも、繰り返し部分の検出は問題なく行われる。また、ＭＥＴＡ（ＰＡＲＴ，ｘ）は、検出のために挿入されたものであるから、最終的に圧縮記述されたファイル中に存在していなくても演奏の再生に対して、全く問題がない。

【００３２】次に本発明の演奏情報圧縮方法の第３実施例について説明する。前述の手順によりＭＥＴＡ（ＰＡＲＴ，ｘ）が挿入されたＳＭＦをＳＭＦ　Ａとし、ＳＭＦＡを圧縮した圧縮ファイルをＳＭＦ　Ｃとする。なお、この圧縮ファイルＳＭＦＣは、スタンダードＭＩＤＩファイルの形式とは異なるので、ＳＭＦと呼ぶことは必ずしも適当ではないが、ここでは便宜的にＳＭＦ　Ｃとしておく。

【００３３】次にいくつかの定義を行っておく。（ａ）ＰＡＲＴ　ＬＩＳＴ（パートリスト）：パート番号の列である。

（ｂ）$f_p$Ｃ（ファイルポインタ）：圧縮ファイルＳＭＦ　Ｃの内容の任意の位置を差し示すもの。

【００３４】ジャンプマークＪとタグＴはＭＥＴＡイベントを用いて次のように記述する。

ＭＥＴＡ（ＪＵＭＰ，Ｃｈ，ｙ）
ＭＥＴＡ（ＴＡＧ，Ｃｈ，ｚ）

【００３５】これらを一組として圧縮ファイルＳＭＦＣに次のように記述する。

ＭＥＴＡ（ＪＵＭＰ，Ｃｈ，ｙ）→△ｔ：ＭＥＴＡ　ＩＤ：ＪＵＭＰ　ＩＤ：ＭＩＤＩチャンネル　Ｃｈ：レベルｙ

ＭＥＴＡ（ＴＡＧ，Ｃｈ，ｚ）→△ｔ：ＭＥＴＡ　ＩＤ：ＴＡＧ　ＩＤ：ＭＩＤＩチャンネル　Ｃｈ：レベルｚ

【００３６】なお、ジャンプマークとタグのそれぞれにレベルを付加することによって、複雑な演奏手順に対しても、統一的な書式で記述し、正しく再生することができる。

【００３７】ここで、ＭＩＤＩチャンネルについて説明する。ＭＩＤＩでは、複数の楽器が同時に演奏されている情報を伝送できるように、ＭＩＤＩのイベントに対して、ＭＩＤＩチャンネル（Ｃｈ）が付加されている。例えば、ピアノに関するイベントはＣｈ＝１、ベースはＣｈ＝２とし、受信側ではチャンネルによってイベントを分離し、それぞれを適切な音源へ転送することがなされ

ている。この概念は、ＳＭＦにおいても全く同様である。楽器毎には、別の演奏を行うも、ＳＭＦ上では、混在しているため繰り返しの検出においてはＳＭＦ　Ａ中のイベントをチャンネル毎に分離し、各チャンネル独立に圧縮記述を行い、この後に、最終的なＳＭＦ　Ｃへ、合成しながら、記録することが望ましい。合成後にも、各チャンネルのジャンプマークとタグが識別できるように、各イベントにＭＩＤＩチャンネルを持つのである。本明細書では説明の簡略のために、ＳＭＦ　Ａは単一のチャンネルのイベントのみを含むものとして説明している。

【００３８】次に第３実施例の圧縮手順についてフローチャートを示す図８及び図９を用いて説明する。圧縮手順のフローには２つのパスがある。１つめのパスは図８に示すように、ＳＭＦ　Ａ中の同一パートを検出し、ＭＥＴＡ（ＰＡＲＴ，ｘ）の書換を行うものである。２つのパートを同じものと見なすために、比較を行う段階でＭＩＤＩフィルタリング、ＭＩＤＩクォンタイズを用いてもよい。これは、ＳＭＦの内容が演奏者の実演奏等を記録したものであるとき、特に効果がある。すなわち、同一パートを演奏しようとした場合でも、実演奏では全く同じＭＩＤＩイベント群にはなり得ないため、ＭＩＤＩフィルタリング、ＭＩＤＩクォンタイズを行い、同一パートの検出率を向上させ、ひいては圧縮率を向上させるものである。ＭＩＤＩフィルタリングは、演奏者が意図しない、極めて短い音に対応するＭＩＤＩイベントを削除すること等である。ＭＩＤＩクォンタイズは発音の時間的な揺れを、大きなタイムスロットに量子化し直すことによって、検出率を向上させるものである。また、音量に対しても同様のことが適用できる。上記は、比較の際に用いる手法であり、最終的に生成されるファイルの内容に対しては用いられないのは当然である。２つ目のパスは図９に示すように、ＳＭＦ　Ａから、ＳＭＦ　Ｃを生成するものである。

【００３９】図８中“ｊ＋＋”“ｉ＋＋”は変数“ｊ”“ｉ”をインクリメントするステップである。変数ｉは比較する元のパートを示し、変数ｊは比較する対象のパートを示している。このフローではパートｉとパートｊの内容が一致したとき、繰り返しがあると判断しＭＥＴＡ（ＰＡＲＴ，ｊ）をＭＥＴＡ（ＰＡＲＴ，ｉ）に変更している。

【００４０】次に図９に従って具体的圧縮手法について説明する。この第３実施例では先の第２実施例で説明した図１６に示すファイルＢに相当するものは用いる必要がない。すなわち、図１６のファイルＡに相当するＳＭＦ　Ａからファイルを用いることなく最終的に圧縮された圧縮ファイルＳＭＦ　Ｃを得るものである。

【００４１】ステップＳ２１で当初内容のないファイルＣ（ＳＭＦ　Ｃ）を作成し、ファイルＣ内の位置を示すファイルＣのポインタをファイルＣの先頭に設定し、当初内容のないパートリストを作成する。次のステップＳ２２で変数ｉにファイルＡの最初のパート番号を設定し、次のステップＳ２３で変数ｉが前記パートリストにあるか否かを判断する。変数ｉで示されるパート番号がないときは、ポインタｆｐＣがファイルＣの末尾にあるか否かをステップＳ２４で判断し、ポインタｆｐＣ位置が前記ファイルＣの末尾以外のときは、ステップＳ２５でポインタｆｐＣが現在ある位置にジャンプマーク（ｎ）を挿入し、次にポインタｆｐＣをファイルＣの末尾へ移動させ、その後ポインタｆｐＣの位置にタグＴＡＧ（ｎ）を挿入し、変数ｎを１つインクリメントする。ポインタｆｐＣ位置がファイルＣの末尾であるときは、ステップＳ２６でファイルＣに前記ファイルＡの変数ｉで示されるパートの情報を付加し、パートリストに変数ｉを付加し、ポインタを前記ファイルＣの末尾に移動させる。

【００４２】パート番号の確認ステップで変数ｉで示されるパート番号があるときは、ポインタｆｐＣがファイルＣの変数ｉで示されるパートの先頭にあるか否かをステップＳ２９で判断する。次のステップＳ３０でポインタｆｐＣの位置が前記ファイルＣの変数ｉで示されるパートの先頭以外のときは、ポインタｆｐＣが現在ある位置にジャンプマークＪＵＭＰ（ｎ）を挿入し、次にポインタｆｐＣをファイルＣの変数ｉで示されるパートの先頭位置へ移動させ、その後ポインタｆｐＣの位置にタグＴＡＧ（ｎ）を挿入し、変数ｎを１つインクリメントする。ポインタｆｐＣが前記ファイルＣの変数ｉで示されるパートの先頭であるときは、ステップＳ３１でポインタｆｐＣを次のパートの先頭位置へ移動させる。この２つの移動させるステップＳ２８又はＳ３１の一方の後に、ステップＳ３２でファイルＡの中の次のパートを検索し、存在する場合は、変数ｉにファイルＡの次のパート番号を設定する。以後、ファイルＡ中に次のパートがなくなるまで、パート番号確認ステップＳ２３以降、次パート番号設定ステップＳ３２までを繰り返し実行する。

【００４３】図１０は図８に示すパス１の開始前のＳＭＦ　Ａと、その終了後に得られるＳＭＦ　Ａを、さらに図９に示すパス２の終了後に得られるＳＭＦ　Ｃを示す図である。図中、Ａ、Ｂ、Ｃ…は各パートの内容を示し、数字１、２、３…はパート番号を示している。またＳＭＦ　Ｃはジャンプマークとタグの付加された様子を示している。

【００４４】次に上記各実施例により圧縮された演奏情報を、例えば通信カラオケの端末側等で再生するために伸張する方法について説明する。図１１は本発明の演奏情報伸張方法を実現するために用いられる演奏情報再生装置（伸張装置）の実施例を示すブロック図である。記憶装置１０は例えば大容量ハードディスクで、この中に図１０に示す圧縮ファイルＳＭＦ　Ｃが多数格納されて

いるものとする。このうち所望の曲のＳＭＦ　Ｃが図示しない制御装置のアドレス指示に従って読み出され、メモリ１２にロードされる。この後、再生に先立ち、ＳＭＦ　Ｃ中のタグのサーチが行われる。すなわち、リーダー１４は、アドレスカウンタ１６の指示に従ってメモリ１２の内容を読んで行き、タグのサーチを行う。分離器１８は、リーダー１４の出力信号、すなわちＳＭＦ　ＣからＭＩＤＩイベント、ジャンプレベル値、タグレベル値を分離抽出するものである。サーチの結果、タグを検出する毎に、タグアドレスメモリ２６は分離器１８から与えられるタグレベル値と、アドレスカウンタ１６から与えられるアドレスを関係付けて図１２に示すテーブルのような形で格納して行く。こうして、１つのＳＭＦ　Ｃを全部サーチして図１２に示すテーブルを完成させる。

【００４５】その後、再生を開始するために、まずアドレスカウンタ１６とレベルカウンタ２２を初期化する。再生時には、分離器１８からのタグレベル値はタグアドレスメモリ２６に書込まない（又は分離器１８からタグレベル値を出力しない）。分離器１８から出力されるＭＩＤＩイベントは各対応する（チャンネル毎の）図示しない音源に与えられる。分離器１８の出力信号の１つとしてレベル比較器２０にはジャンプレベル値が与えられる。レベルカウンタ２２はそのカウント値をレベル比較器２０へ出力し、レベル比較器２０は２つの入力レベルを比較し、一致するとレベルカウンタ２２を１つインクリメントさせる信号を出力するとともにゲート２４へゲート信号を与える。ゲート２４はレベルカウンタ２２の出力カウント値、すなわち、検出されたジャンプマークのレベル値（一致レベル値）をゲート信号を受けたとき、選択器２８へ与える。選択器２８はタグアドレスメモリ２６からジャンプレベルに対応したタグのアドレスを読み出して、アドレスカウンタ１６にロードする。アドレスカウンタ１６は、ロードされたアドレスに従って順次メモリ１２内のデータを読み出して行く。

【００４６】このようにして、メモリ１２内の１つのＳＭＦ　Ｃの内容を読みながらＭＩＤＩイベントを順次出力することにより伸張動作が行われ、ＳＭＦ　Ｃの全てのデータを読み出すことにより一曲分の再生を終了する。

【００４７】図１１に示した構成の上記説明では、再生に先立ってタグの位置をサーチして、そのアドレスを予めロードする方法を用いているが、伸張装置の処理能力が十分なものであれば、予めタグを読み込まず、タグをサーチしながら再生を同時に行うようにすることもできる。かかる構成の場合は、図１１に示したタグアドレスメモリ２６は不要となる。

【００４８】

【発明の効果】以上詳細に説明したように、本発明の演奏情報圧縮方法によれば、ＭＩＤＩ信号のように楽音を構成する音の音程と発音時間を指定する演奏情報のデータ量を、メロディーの繰り返しを削除することにより、大幅に削減することができ、また本発明の演奏情報伸張方法によれば、かかる削減により圧縮された演奏情報を伸張して元の繰り返しのある演奏情報を再生することができ、よって通信コストの削減、メモリ容量の問題の解消がなされる。また、通信カラオケシステム等において、画像情報や文字情報を本来の演奏情報信号とともに送信したり、メモリに蓄積する場合に、演奏情報のデータ量が削減できるので、これらの付加的情報の取り扱いが容易となる。

【図面の簡単な説明】

【図１】本発明の演奏情報圧縮方法の第１実施例を示すフローチャートである。

【図２】本発明の演奏情報圧縮方法及び演奏情報伸張方法の適用される一例としての通信カラオケシステムのブロック図である。

【図３】本発明の演奏情報圧縮方法及び演奏情報伸張方法の適用される一例としての通信カラオケシステムの他の構成例を示すブロック図である。

【図４】本発明の演奏情報圧縮方法の概念を示すブロック図である。

【図５】本発明の演奏情報圧縮方法の手法を説明する図である。

【図６】本発明の演奏情報圧縮方法の第１実施例で用いられる信号のフォーマット図である。

【図７】本発明の演奏情報圧縮方法の第１実施例で圧縮しようとする曲の一部の楽譜である。

【図８】本発明の演奏情報圧縮方法の第３実施例の一部を示すフローチャートである。

【図９】本発明の演奏情報圧縮方法の第３実施例の他の一部を示すフローチャートである。

【図１０】本発明の演奏情報圧縮方法の第３実施例における圧縮ファイル作成過程を説明する図である。

【図１１】本発明の演奏情報伸張方法を実現する再生器の一例を示すブロック図である。

【図１２】図１１におけるタグアドレスメモリの内容を示す図である。

【図１３】本発明の演奏情報圧縮方法の第２実施例におけるジャンプマークとタグのデータ構成を示す図である。

【図１４】本発明の演奏情報圧縮方法の第２実施例の圧縮手順を説明する図である。

【図１５】本発明の演奏情報圧縮方法の第２実施例の圧縮手順を説明する図である。

【図１６】本発明の演奏情報圧縮方法の第２実施例の圧縮手順を説明する図である。

【図１７】本発明の演奏情報圧縮方法の第２実施例の圧縮手順を説明する図である。

【図１８】本発明の演奏情報圧縮方法の第２実施例の圧

19

縮手順を説明する図である。

【図１９】本発明の演奏情報圧縮方法の第２実施例の圧縮手順を説明する図である。

【図２０】本発明の演奏情報圧縮方法の第２実施例の圧縮手順を説明する図である。

【図２１】本発明の演奏情報圧縮方法の第２実施例の圧縮手順を説明する図である。

【図２２】本発明の演奏情報圧縮方法の第２実施例の圧縮手順を説明する図である。

【図２３】本発明の演奏情報圧縮方法の圧縮前のＳＭＦを模式的に示す図である。

【図２４】本発明の演奏情報圧縮方法の図２３のＳＭＦを圧縮する１つの手法を説明する図である。

【符号の説明】

１　ＳＭＦ（スタンダードＭＩＤＩファイル）

20

２　ＭＩＤＩチャンネル分離ブロック

３　ブロックリピート検出ブロック

４　Ｒ（リピート）符号化処理ブロック

５　マージ処理ブロック

６　圧縮ファイル

１０　ＳＭＦ　Ｃの記憶装置

１２　メモリ

１４　リーダー

１６　アドレスカウンタ

１８　分離器

２０　レベル比較器

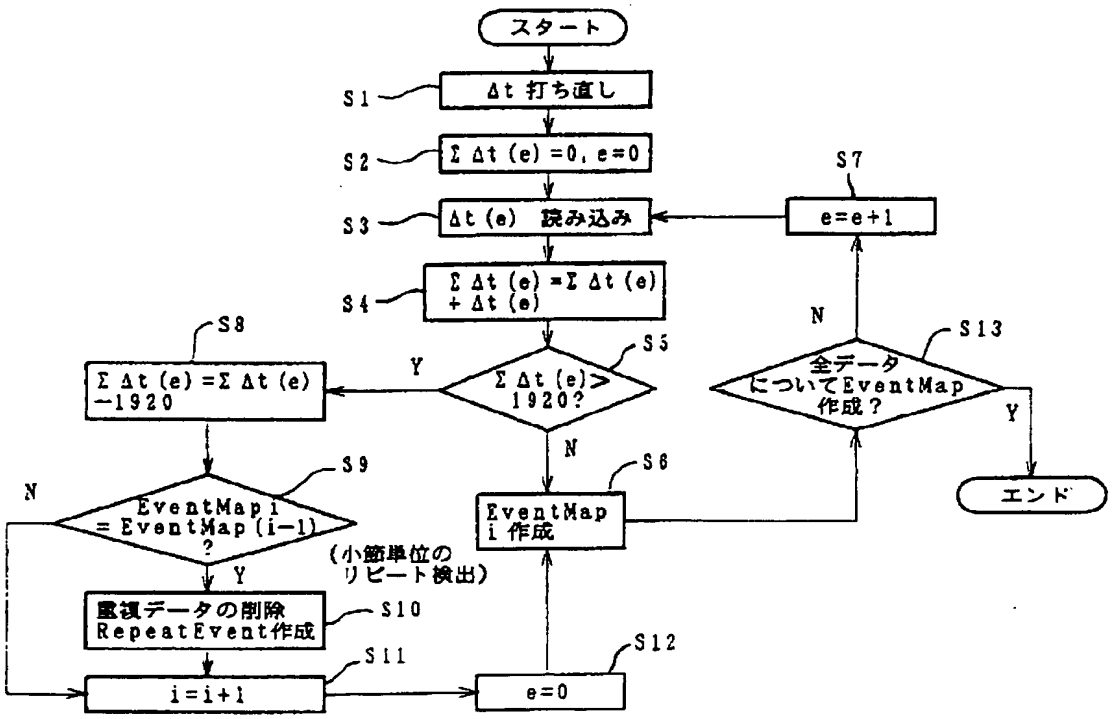
２２　レベルカウンタ

２４　ゲート

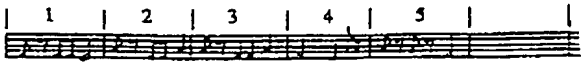
２６　タグアドレスメモリ

２８　選択器

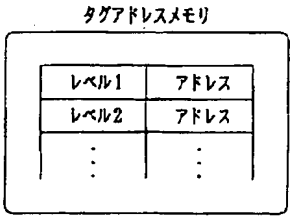
【図１】



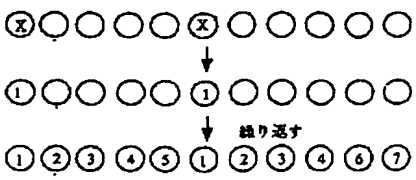

【図７】



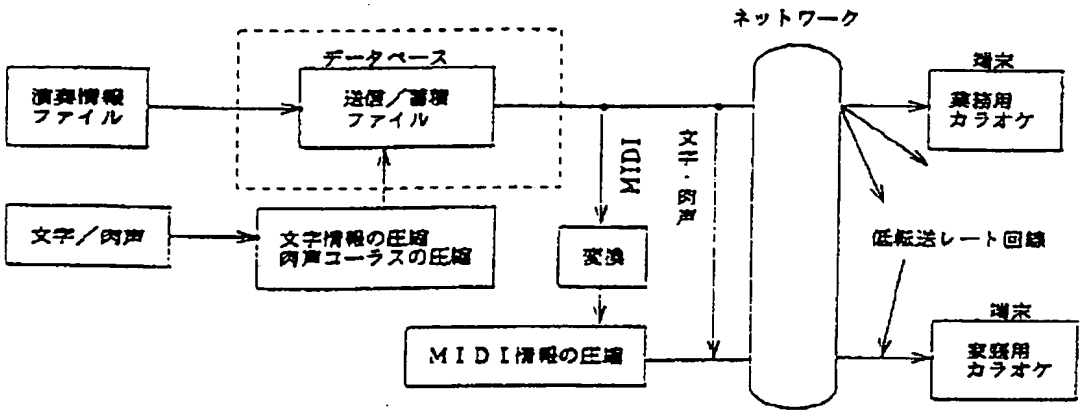

【図１２】



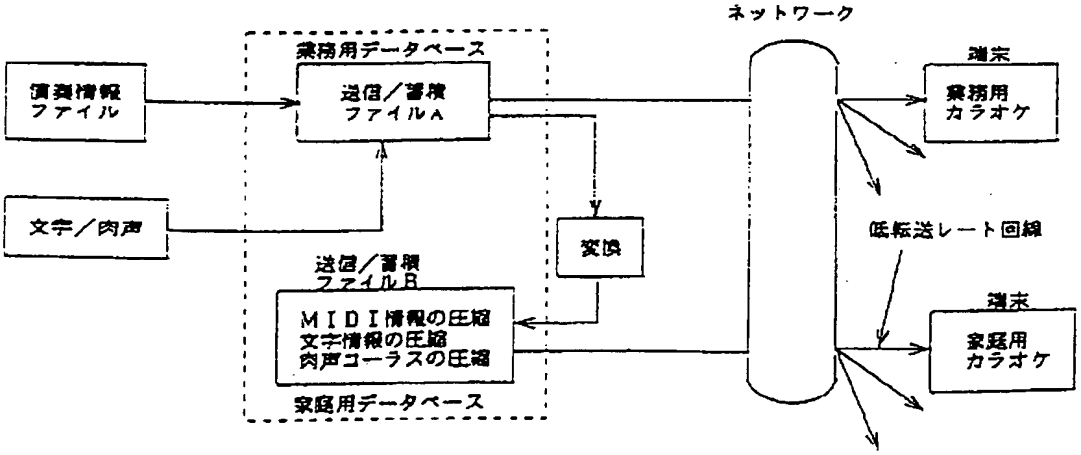

【図１５】



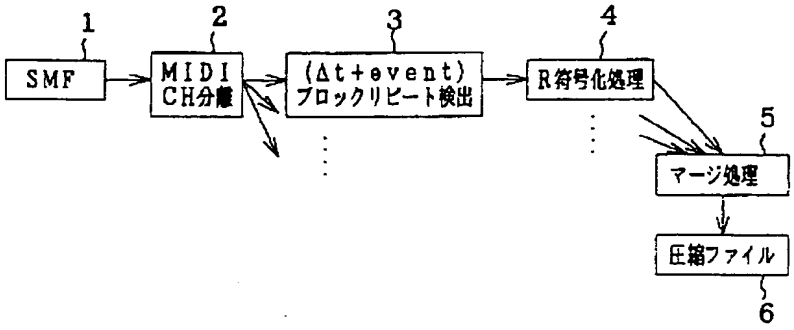
【図２】



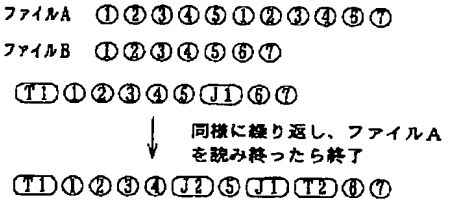

【図３】



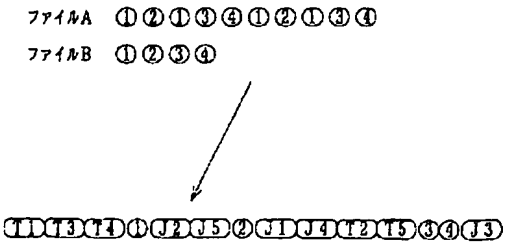

【図４】



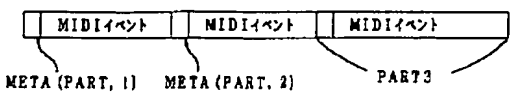

【図１６】




【図１８】




【図１９】



【図５】

SMFによる記述

Δt1 EVENT1 Δt2 EVENT2 Δt1 EVENT1 Δt2 EVENT2 Δt3 EVENT

(前の2バイトと同一演奏の繰返し)

↓

本圧縮法方式による記述

T(Δt1と同定義) EVENT1 T(Δt2と同定義) EVENT2 R T(Δt3と同定義) EVENT3

(repeat)

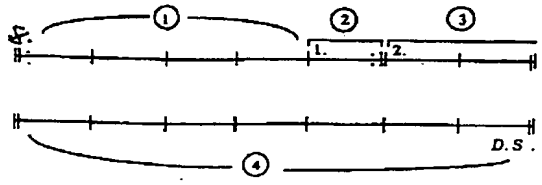
【図６】

T又はRのフォーマットデータ構造(SMFのΔt形式を変更する)

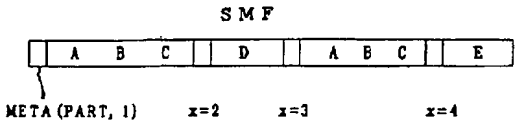
| T又はR | ID1 | ID0 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|

(ID1,ID0)=(0,0)　:Δt(従来と同じ)
D5　:後続バイトフラグ
D4～D0　:Ticks値(デルタタイムの最上位バイト(00-3F。SMFの場合は00-7F)を記述。後続バイトがある場合(D5=1の場合)には、2バイト目は40－1FFF(SMFの場合は80-3FFF)となる)
(ID1,ID0)=(0,1)　:拡張Δt(指定された繰返し回のみ演奏を行う為のΔt)
D5～D1　:演奏回指定(その回のみ演奏)
D0　:後続バイトフラグ(0ならばΔt=0とする)
(ID1,ID0)=(1,0)　:単純リピート(直前の指定区間を繰返し)
D5～D0　:リピート回数(何回繰り返すかを指定)
第2バイト　:-Δtの値(現在位置から何Ticks前までを繰り返すかを指定)
(ID1,ID0)=(1,1):　区間指定リピート(指定された区間を繰返し)
D5～D0　:リピート回数(何回繰り返すかを指定(1回～64回))
第2バイト　:-Δtsの値(現在位置から何Ticks前から繰り返すかを指定。
第3バイト　:-Δteの値(現在位置から何Ticks前迄繰り返すかを指定。
(第2バイトは可変長のため、-Δteは3バイト目に記述されるとは限らない)

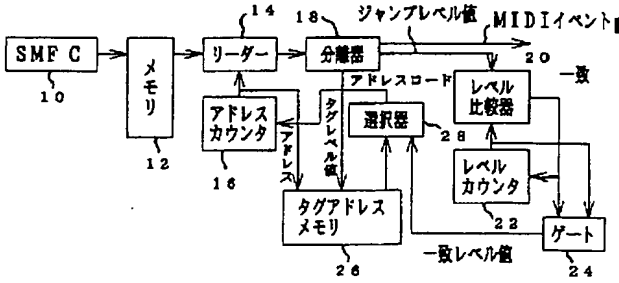
【図１７】



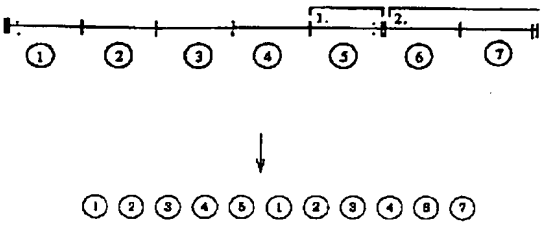

【図２１】




【図１１】



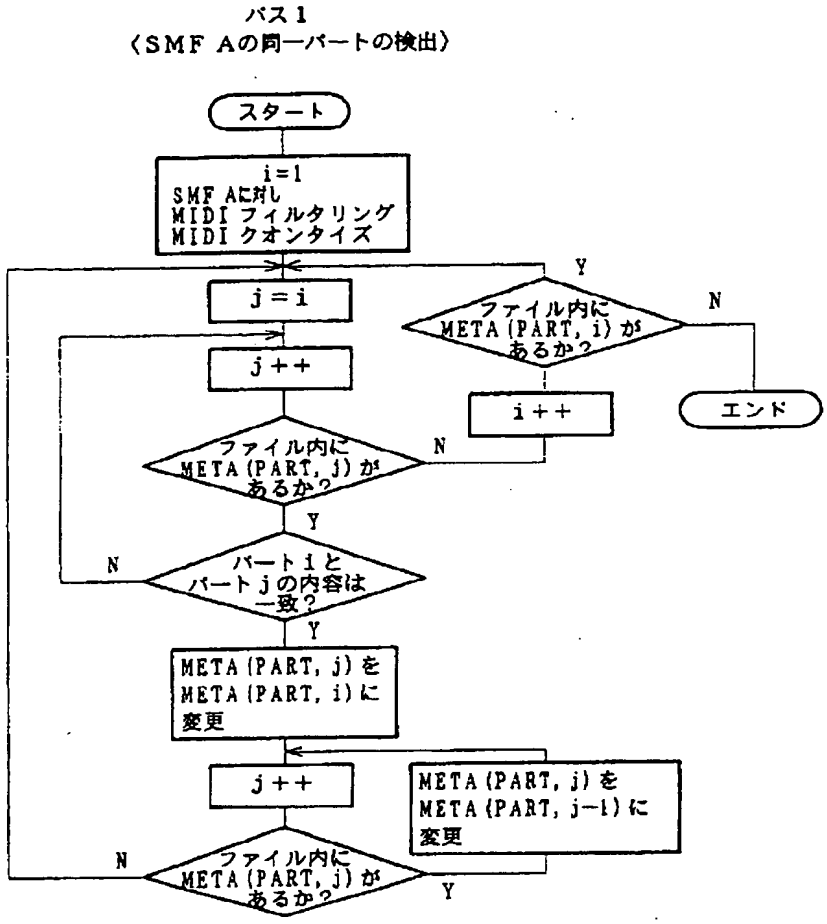

【図１４】



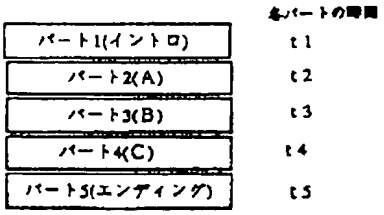

【図１３】

| | | | | |
|---|---|---|---|---|
| ジャンプマーク | 非演奏情報ＩＤ | マークＩＤ | チャンネルＩＤ | レベルＩＤ |
| タグ | 非演奏情報ＩＤ | タグＩＤ | チャンネルＩＤ | レベルＩＤ |

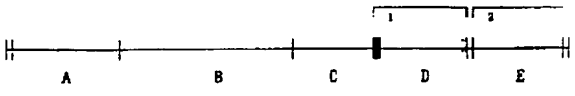
【図８】



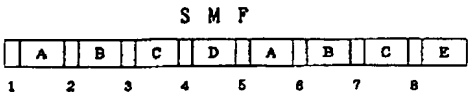

【図２４】



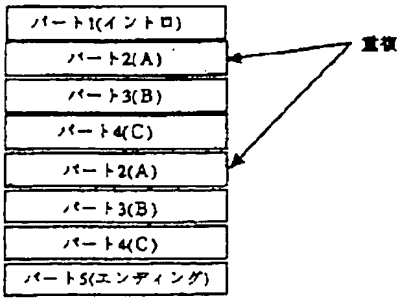

【図２０】



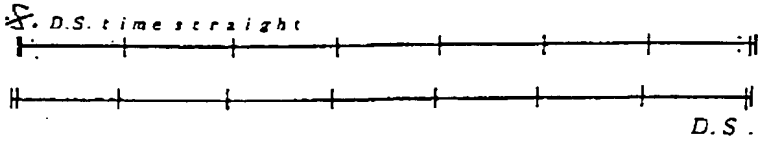

【図２２】



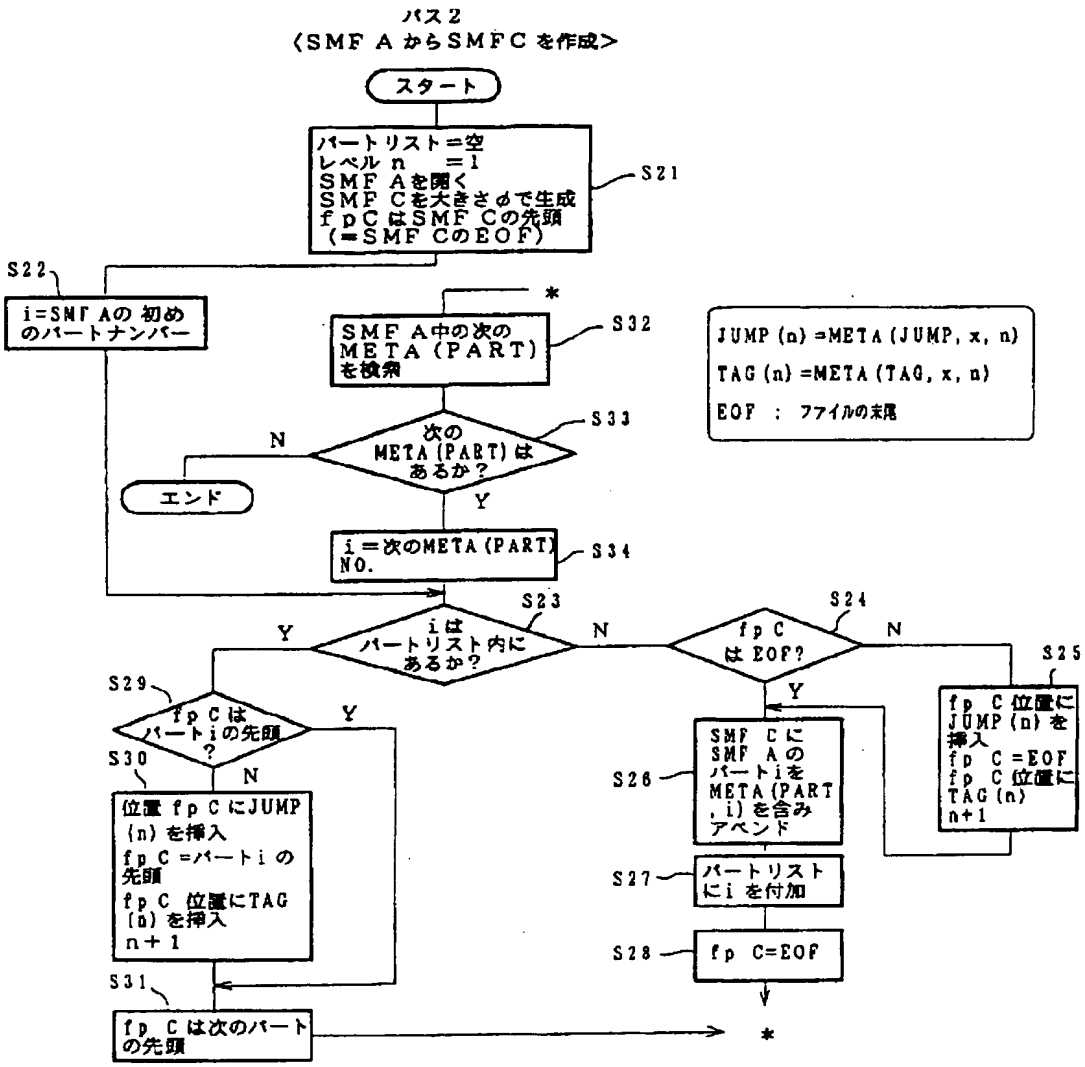

【図２３】



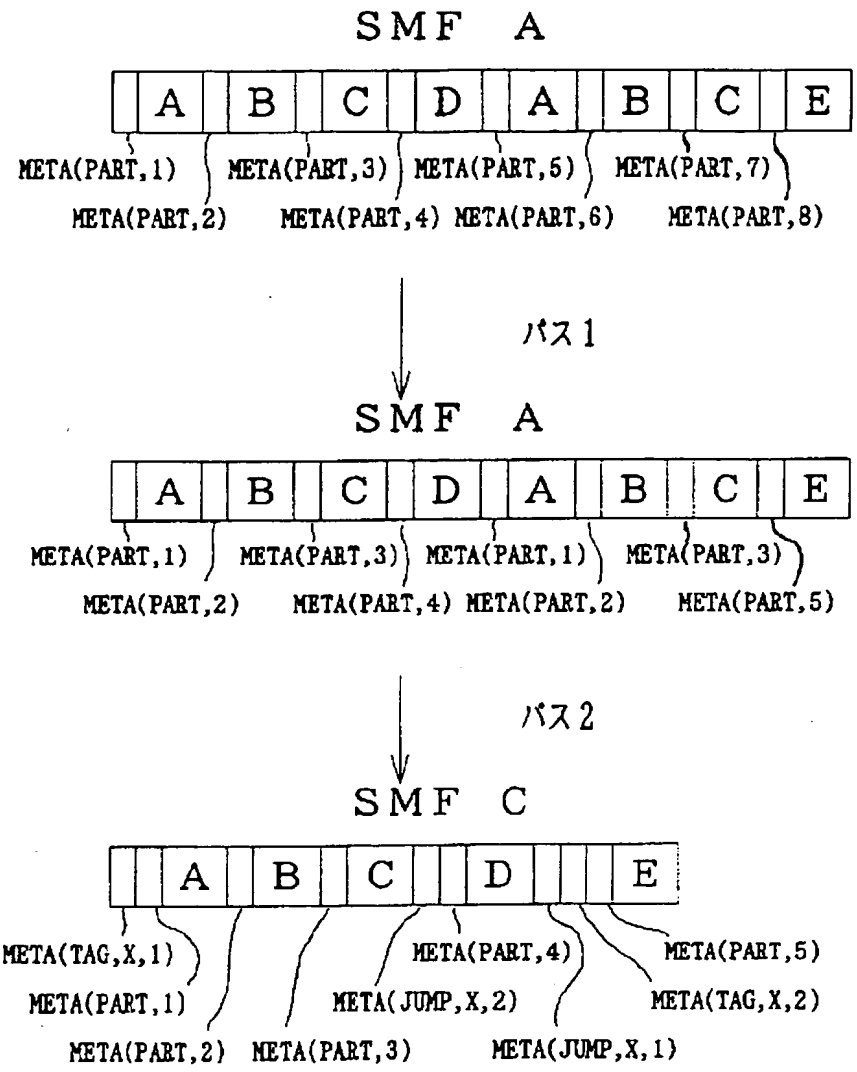

【図２５】



【図９】


パス２
〈SMF AからSMFCを作成〉
スタート
パートリスト＝空
レベル n ＝1
SMF Aを開く
SMF Cを大きさ0で生成
fpCはSMF Cの先頭
(＝SMF CのEOF)
S21
S22
i=SMF Aの 初め のパートナンバー
*
SMF A中の次の META (PART) を検索
S32
JUMP (n) =META (JUMP, x, n)
TAG (n) =META (TAG, x, n)
EOF : ファイルの末尾
次の META (PART) は あるか？
S33
N
Y
エンド
i＝次のMETA (PART) NO.
S34
S23
iは パートリスト内に あるか？
Y
N
S24
fp C は EOF?
N
Y
S25
fp C位置に JUMP (n) を挿入
fp C =EOF
fp C位置に TAG (n) n+1
S29
fp Cは パートiの先頭 ？
Y
N
S30
位置 fp C にJUMP (n) を挿入
fp C =パートi の先頭
fp C 位置にTAG (n) を挿入
n＋1
S26
SMF C に SMF A の パートiを META (PART ,i) を含み アペンド
S27
パートリスト にi を付加
S28
fp C=EOF
S31
fp C は次のパート の先頭
*

【図１０】




【手続補正書】
【提出日】平成６年１２月１５日
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】図２５
【補正方法】追加
【補正内容】
【図２５】Ｄ．Ｓ．（ダルセーニョ）を含んだ楽譜の例を示す図である。